Phone Contract Exercise

Evaluation Criteria

• Does the Java code provide a complete and accurate solution for the stated requirements? Including, but not limited to the following:

- Do the methods do what they are expected to do?
- Is the right data returned in the right order?
- Does the DAO follow the normal practices for that pattern as taught on the course?

• Do the JUnit tests provide adequate coverage of the functionality? Are exceptions used to manage errors that can occur in the code? For example:

- Are there enough tests to fully cover the functionality?
- Do the tests follow JUnit and TDD best practices?
- Are there incorrect results that would pass the tests?
- Are the tests self-contained and repeatable?
- Does exception handling follow best practices?
- Is exception handling tested as far as possible?

- Does the Java code follow best practices? For example: o Well laid out and easy to read
  - Short methods with descriptive names
  - DRY – Don't Repeat Yourself
  - YAGNI – You Ain't Going to Need It
  - SOLID principles
  - PIE

- Does the SQL meet the requirements? Does it follow best practices? For example: o Are the correct results returned in the right order?
  - Is code well laid out and readable?
  - If tables must be joined, are they joined on the right columns?

Scenario

You have been assigned to a development team that is managing phone contracts. You have been provided with the following:

A. A database schema and the DDL to create and populate it. You may not change the tables in the schema.

B. A value class that represents the core data in the schema. You may not change the name or data type of any of the fields provided. However, you may create additional methods in this class.

C. An interface that represents a Data Access Object. You may add methods to the interface as long as they are appropriate according to object-oriented best practice, but you may not remove any methods since another team has already started working on components that depend on the interface.

D. Another interface for a service that processes Phone Contracts. You may not make any changes to this interface at all.

Tasks

1. Implement the DAO interface. Note the following additional requirements:

    - All currency amounts must be represented with 2 decimal places.
    - All DAO methods should return fully-populated objects.

2. Implement the service.
    - The method should throw an exception if the list is null.

3. Work TDD.
    - You must provide tests for any functionality that you create.
    - Your tests must be self-contained and repeatable.